```
admin@admin: ~/LLM-rcm$ mdcat title-slide.md
```

#LLM Recommendation

{“Thành viên”: [“Nguyễn Minh An”, “Nguyễn Hồng Phúc”]}

Dữ liệu sử dụng

- Tên dữ liệu: Movielens-20M
- Chứa khoảng 20 triệu lượt đánh giá và 465.564 tags lên 27.278 bộ phim.
- Bộ dữ liệu được thu thập từ 1995 đến 2015 bởi 138.493 người dùng.

Các trường dữ liệu

ratings.csv

Thông tin đánh giá phim của người dùng
- userId: ID của người dùng
- movieId: ID của phim
- rating: Điểm đánh giá (0.5 đến 5.0)
- timestamp: Thời điểm đánh giá

movies.csv

Thông tin chi tiết về phim

- movieId: ID của phim
- title: Tiêu đề phim
- genres: Thể loại phim

tags.csv

Thẻ (tags) do người dùng gắn cho phim
- userId: ID của người dùng
- movieId: ID của phim
- tag: Tên thẻ
- timestamp: Thời điểm gắn thẻ

links.csv

Liên kết phim đến các ID trong IMDb và TMDb

- movieId: ID của phim trong MovieLens
- imdbId: ID trong IMDb
- tmdbId: ID trong TMDb

genome-scores.csv

Điểm liên quan giữa phim và thẻ trong bộ "tag genome"
- movieId: ID của phim
- tagId: ID của thẻ
- relevance: Mức độ liên quan (giá trị từ 0 đến 1)

genome-tags.csv

Danh sách thẻ trong bộ "tag genome"

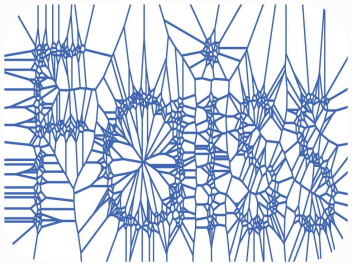
- tagId: ID của thẻ
- tag: Tên thẻ

Công nghệ sử dụng

Thuộc tính	Chi tiết
model code	models/gemini-2.0-flash models/text-embedding-004
rate limits	1500 RPM (embed model) 2000 RPM (generative model)
output dimension	768 (embed model) 8192 (generative model)

LLM embed và generative

Công nghệ sử dụng



FAISS Vector Database

MongoDB

Streamlit

FastAPI

Các phần chính

01 Tiền xử lý

02 Ý tưởng thuật toán

03 Cách 1

04 Cách 2

05 Đánh giá

06 Phát triển & Mở rộng

07 Demo

Dữ liệu về phim

- Tổng hợp và lọc ra những phim có ít nhất **10 tags** có độ phù hợp **> = 0.7**
- Kết hợp với dữ liệu cào thêm từ **OMDB API** (Tên đạo diễn, diễn viên, cốt truyện, ...)
- Áp dụng công thức **bayesian weight average** để tính rating trung bình của từng bộ phim.
- Gộp các thông tin của từng phim lại thành từng chuỗi lớn.

OMDB API

$$s_i = \frac{m_i}{m_i + m_{avg}} \cdot A_i + \frac{m_{avg}}{m_i + m_{avg}} \cdot S$$

Weighted average

=> Kết quả: 10342 phim, 12 trường

Dữ liệu người dùng

- Lọc ra những user đã xem ít nhất **100 phim**.

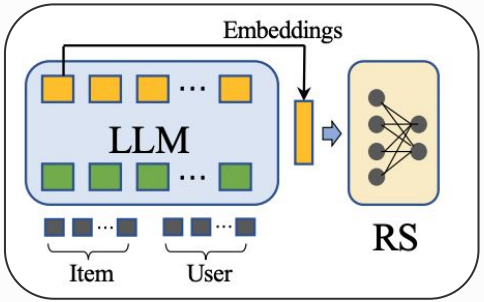
- Sắp xếp các phim đã xem của từng user theo **timestamp**.

=> Thu được 15 triệu đánh giá từ 52366 users

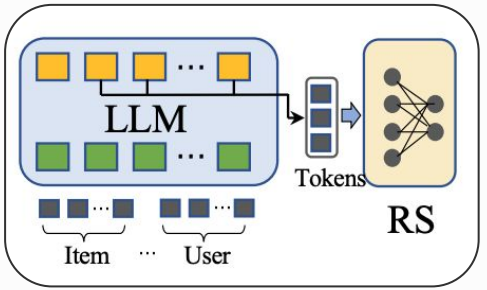
		rating	timestamp
userId	**movieId**		
1	924	3.5	2004-09-10 03:06:38
	919	3.5	2004-09-10 03:07:01
	2683	3.5	2004-09-10 03:07:30
	1584	3.5	2004-09-10 03:07:36
	1079	4.0	2004-09-10 03:07:45
...	...	...	...
138493	6534	3.0	2009-12-07 18:18:28
	53464	4.0	2009-12-07 18:18:40
	1275	3.0	2010-01-01 20:42:32
	6996	3.0	2010-01-01 20:42:35
	405	3.0	2010-01-01 20:42:52

15829284 rows × 2 columns

#Ý tưởng



Cách 1:
Content-based Filtering
(LLM embedding)



Cách 2:
Hybrid Filtering
(Collaborative + Text Summary)

Tạo vector database cho phim

- Thiết lập FAISS vector database với cách thức tìm kiếm theo tích vô hướng với số chiều 768.

- Sử dụng embed model để vector hóa chuỗi lớn của các phim.

- Chuẩn hóa và thêm những vector đó vào trong FAISS vector database và đánh chỉ số index để tái sử dụng.

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

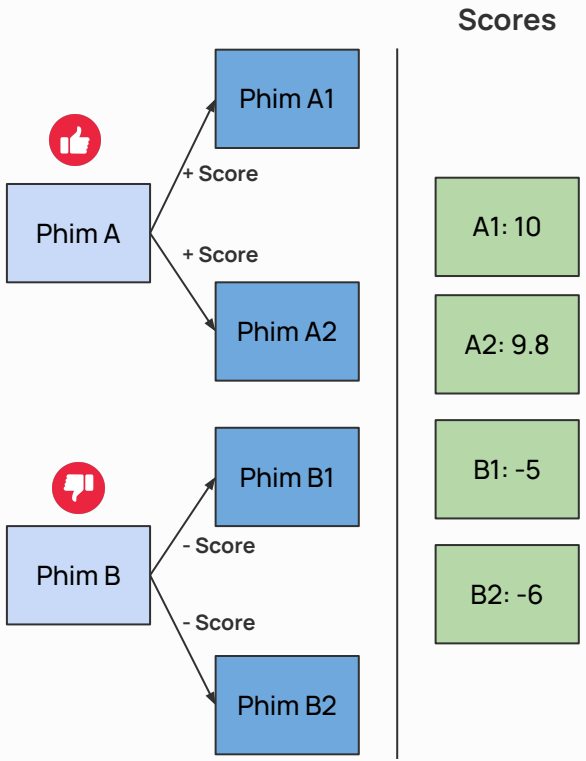
Cosine similarity

Điểm ranking

- Từ lịch sử xem của user, lấy ra n phim giống nhất với mỗi phim lịch sử và tính toán điểm score ranking của n phim đó.

- Nếu n phim lấy ra từ phim mà user thích (>= 3 sao), **“thưởng”** cho n phim đó.

- Nếu n phim lấy ra từ phim mà user ghét (< 3 sao), **“phạt”** n phim đó.



Công thức

- Phụ thuộc vào 3 tham số: user_rate, weight_rating và similarity.

Combine_rating = 0.8 * user_rate(i) + 0.2 * weight_rating(i)

Combine rating cho mỗi phim

- Combine rating sẽ giúp đánh giá bao gồm cả độ nổi tiếng của phim.

Score(j) += combine_rating * similarity(j, i)

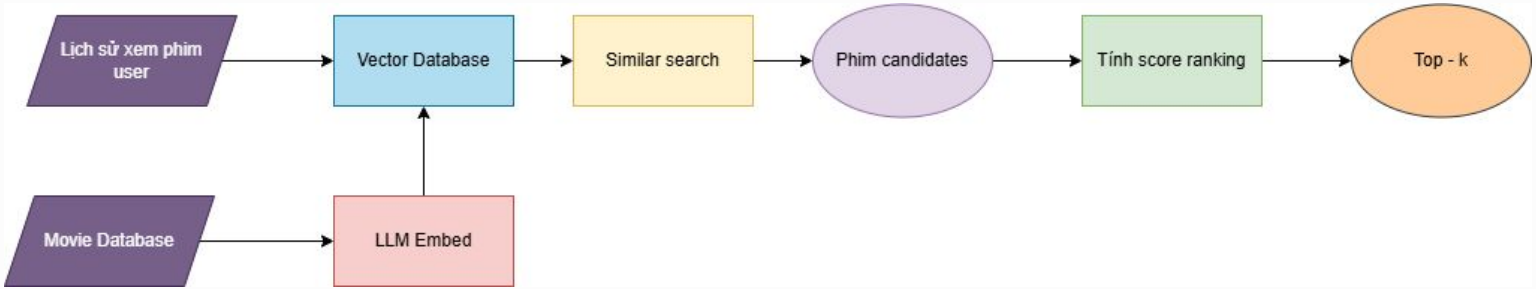
Những phim giống với phim user thích

- Alpha càng lớn thì lượng **"phạt"** càng nhiều.

Score(j) += (combine_rating - alpha * (5 - user_rate(i))) * similarity(j, i)

Những phim giống với phim user ghét

Luồng chạy


Lịch sử xem phim user
Vector Database
Similar search
Phim candidates
Tính score ranking
Top - k
Movie Database
LLM Embed

#Hybrid Filtering

Vấn đề

Content-based Filtering

- Bị giới hạn bởi chất lượng metadata
- Tiêu tốn tài nguyên
- Không tận dụng được kiến thức cộng đồng

Collaborative Filtering

- Cold-start Problem

→ Kết Hợp: **Collaborative Filtering + Content-based Filtering**

Chuỗi mô tả của phim Toy Story (1995) từ cách 1:

"**genres**: Adventure,Animation,Children,Comedy,Fantasy
tags: adventure, animated, animation, cartoon, cgi,childhood, children, classic, clever, computer animation, cute, disney, disney animated feature, entertaining, family, feel-good, friendship, fun, fun movie, good, good soundtrack, great, great movie, heartwarming, imdb top 250, kids, kids and family, light, nostalgic, original, original plot, oscar (best animated feature), oscar winner, pixar, pixar animation, story, storytelling, technology, toys, unlikely friendships, very good, whimsical, witty
plot: A little boy named Andy loves to be in his room, playing with his toys, especially his doll named "Woody". But, what do the toys do when Andy is not with them, they come to life. Woody believes that his life (as a toy) is good. However, he must worry about Andy's family moving, and what Woody does not know is about Andy's birthday party. Woody does not realize that Andy's mother gave him an action figure known as Buzz Lightyear, who does not believe that he is a toy, and quickly becomes Andy's new favorite toy. Woody, who is now consumed with jealousy, tries to get rid of Buzz. Then, both Woody and Buzz are now lost. They must find a way to get back to Andy before he moves without them, but they will have to pass through a ruthless toy killer, Sid Phillips.
actors: Tom Hanks, Tim Allen, Don Rickles
directors: John Lasseter
language: English"

→ Mô tả dài, tốn kém token và có thể không hiệu quả.

#Hybrid Filtering

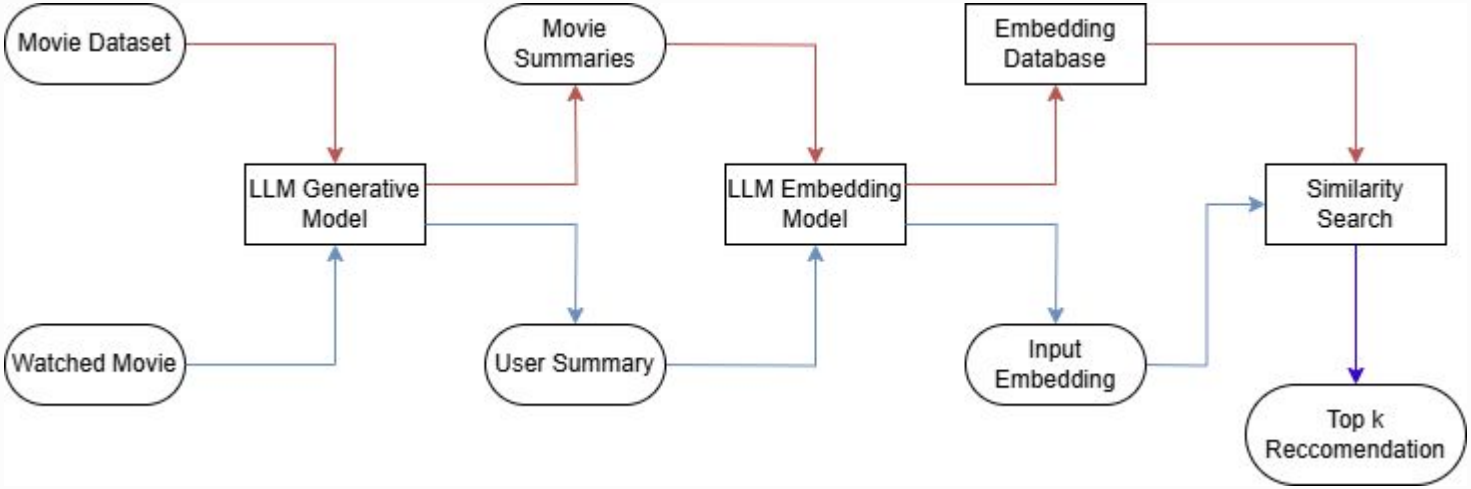
Collaborative Filtering

Collaborative Filtering là một kỹ thuật phổ biến trong hệ thống gợi ý, dựa vào hành vi và sở thích của người dùng thay vì thông tin nội dung của sản phẩm.

- **Model-based** Collaborative Filtering
- Model-based sử dụng kỹ thuật học máy để học các đặc trưng ẩn (latent features) từ dữ liệu đánh giá.
- Model: **Singular Value Decomposition (SVD)**

#Hybrid Filtering

Content-based Filtering (Text Summary)



#Text Summary Method

Movies Summarize

Input: Tập dữ liệu phim (Movie Dataset) chứa các thông tin tiêu đề, thể loại, tags.
Processing: Dữ liệu này được đưa vào mô hình LLM Generative Model thông qua prompt:

```
“Analyze these movies and summarize each in this EXACT format:
Genres: [genres] | Themes: [3-5 themes] | Style: [2-3 style words] | Notable: [1-3
standout elements]

Input Data:
      <Movie Data>

Provide exactly <number of movie> summaries, one per line:”
```

Output: Mỗi phim có một 1 câu Movie Summary dạng văn bản tự nhiên mô tả thể loại, chủ đề, phong cách và yếu tố nổi bật của phim
→ Mô tả ngắn gọn, súc tích, khái quát được bộ phim và tối ưu cho việc tìm kiếm độ tương đồng

#Text Summary Method

User Summarize

Input: 100 bộ phim đã xem gần nhất của người dùng (Watched Movies)
Processing: Các movie summaries tương ứng với các phim đã xem được gom lại và đưa vào LLM
LLM Generative Model thông qua prompt:

```
“Analyze the following movie ratings and summaries to infer this user's preferences
EXACTLY this format:
Genres: [5-6 favorite genres] | Themes: [4-5 favorite themes] | Style: [3-4 stylistic
preferences] | Notable: [2-3 standout elements they seem to enjoy]

Rated Movies:
      <Watched Movies Summaries, Ratings>

Based on the above, write a one-line summary of the user's preferences.”
```

Output: Một câu văn bản tổng hợp ngữ nghĩa mô tả thể loại, chủ đề, phong cách và yếu tố nổi bật mà người dùng yêu thích.

#Text Summary Method

Recommendation

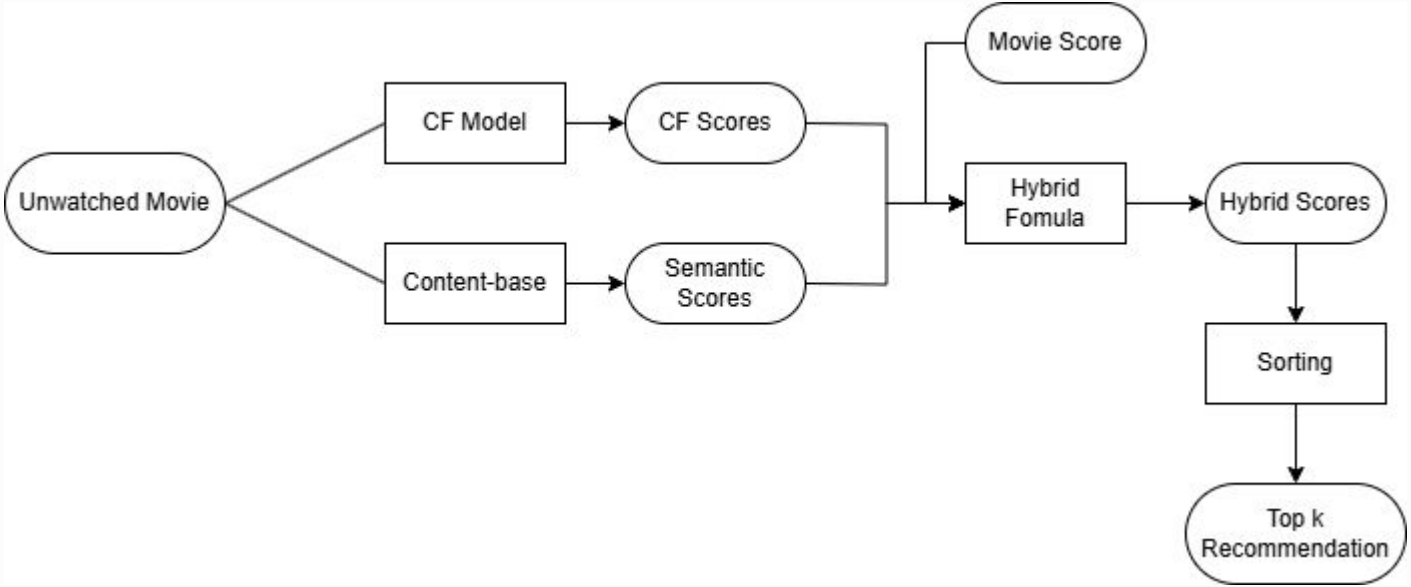
- Các Movie Summaries được embed bằng LLM Embedding Model và lưu vào Embedding Database FAISS.
- Với đầu vào là lịch sử xem của một người dùng thì sẽ được tóm tắt lại, chuyển hóa thành vector và thực hiện phép so sánh cosine để trả lấy ra k bộ phim có điểm số cosin cao nhất.

Ưu điểm:

- Hiểu sâu hơn về nội dung phim và sở thích người dùng nhờ sử dụng LLM.
- Giải quyết vấn đề cold-start.
- Tính cá nhân hóa cao.

#Hybrid Filtering

Collaborative Filtering + Text Summary



#Hybrid Filtering

Collaborative Filtering + Text Summary

Công thức:

$$\text{Hybrid Score} = 0.5 \times \text{CF Score} + 0.3 \times \text{Semantic Score} + 0.2 \times \text{Movie Score}$$

- **CF Score**: Rating dự đoán từ mô hình **SVD** (0 - 1)
- **Semantic Score**: Điểm tương đồng cosin từ phương pháp text summary (0 - 1)
- **Movie Score**: Điểm của phim (0 - 1)

$$\text{Movie Score} = 0.4 \times \text{Weighted Average} + 0.6 \times \text{Popularity Score}$$

Trong đó:

- **Weighted Average**: Điểm chất lượng của phim (0 - 1)
- **Popularity Score**: Điểm của độ nổi tiếng/hot của phim (0 - 1)

#Hybrid Filtering

Ý nghĩa

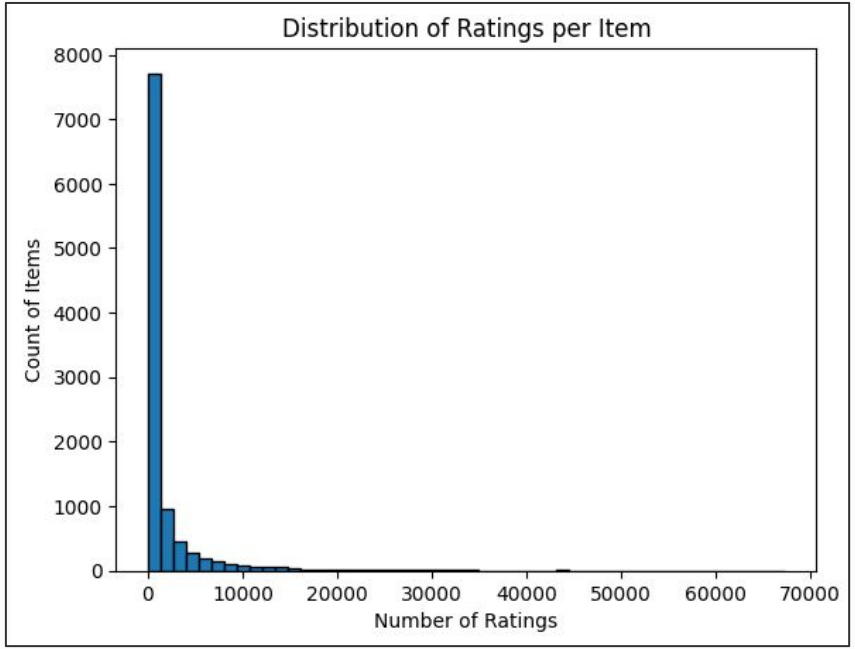
- Tận dụng triệt để sức mạnh tổng quát hóa của **LLM**: khả năng cá nhân hóa, hiểu ngữ cảnh và tổng hợp nội dung.

- Kết quả gợi ý dựa trên đa dạng tiêu chí:
  - Thị hiếu cộng đồng
  - Chất lượng bộ phim
  - Độ tương thích với sở thích người dùng
  - Độ nổi tiếng

Cách đánh giá

- Mỗi user đều đánh giá phim khác nhau và số lượng cũng khác nhau
- Không phải phim nào cũng được mọi user đánh giá

=› Cần phải đánh giá theo từng user sau đó tính trung bình



Đánh giá

Chuẩn bị dữ liệu

- Tất cả các phim của user được sắp xếp theo timestamp.
- Chỉ giữ lại những user đã rate ít nhất 100 phim.
- Train, test được chia theo tỉ lệ 75/25 với mỗi user.
- Số lượng user được đánh giá: 115.

Đánh giá

	HR@10	HR@20
Cách 1	46.2	63.1
Cách 2	43.5	57.4
Cách 2 + MS	60.9	70.4

- **Cách 1**: Phương pháp **LLM Embedding** cho Content-base.
- **Cách 2**: **CF** kết hợp với **TS** với tỷ lệ 70/30.
- **Cách 2 + MS**: **CF + TS** kết hợp thêm Movies Score.

```
admin@admin: ~/LLM-rcm$ mdcat demo.md
```

#Demo

#Phát triển

- Cải thiện, sử dụng model LLM khác mạnh hơn.
- Tối ưu prompt, trích xuất nhiều thông tin hữu ích hơn cho việc gợi ý.

```
admin@admin: ~/LLM-rcm$ mdcat last-goodbye.md
```

#ARIGATHANKS!